REMARKS

I INTRODUCTION

Claims 1-3, 5, 8-10, 13-23, and 25-27 remain pending in the present application. Claims 1, 8, and 15 have been amended to more particularly point out and distinctly claim the present invention. No new matter added has been added. In view of the above amendments and the following remarks, it is respectfully submitted that all of the presently pending claims are allowable.

II THE 35 U.S.C. §103 REJECTIONS SHOULD BE WITHDRAWN

Claims 1-3, 5, 8-10, 13-23, and 25-27 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 5,359,721 to Kempf et al. ("Kempf") in view of U.S. Patent No. 4,430,705 to Cannavino et al. ("Cannavino").

Kempf describes a method of allowing a process executing in non-supervisor mode to perform dynamic linking across address spaces without compromising system security. (*See* Kempf, Col. 2, ll. 50-52). Kempf compares this to a typical process executing in non-supervisor mode, which cannot access another process executing in another address space without invoking the operating system. (*See* Kempf, Col. 1, ll. 24-27). Although a process can dynamically link shared libraries into itself when it starts up without entering supervisor mode, security may be compromised. (*See* Kempf, Col. 2, ll. 23-29).

Cannavino describes an authorization mechanism for establishing addressability to information in another address space. This mechanism permits a program in one address space to access data in another address space without invoking a supervisor. (*See* Cannavino, Abstract). A subsystem control facility provides basic authority control with dual address space memory references, program subsystem linkages, and Address Space Number ("ASN") translation to main memory addresses with authorization control. (*See* Cannavino, Col. 3, ll. 9-14). Basic authority control makes a level of privilege or authority available to problem programs. (*See* Cannavino, Col. 3, ll. 15-16). The dual address space concept provides the problem program

with the ability to move information from one address space into another, and also to specify in which address space the operands of the program are to be accessed. (*See* Cannavino, Col. 3, ll. 53-57). This incorporates a segment table, which is used to translate virtual addresses of a particular address space. (*See* Cannavino, Col. 3, ll. 57-64). The subsystem linkage control provides for direct linkage between problem programs by authorizing the execution of a Program Call and Program Transfer instruction. (*See* Cannavino, Col. 4, ll. 17-24). The ASN feature provides translation tables and authorization controls whereby a program in the problem state can designate an address space as being a primary address space or a secondary address space. (*See* Cannavino, Col. 4, ll. 66 - Col. 5, ll. 2).

Claim 1 has been amended to recite a method comprising "loading a code module into a memory space of a first domain, *wherein the first domain owns one of a kernel space and a portion of a user space*, the code module including an instruction having a symbol reference; determining if the symbol reference is to an external location outside of the memory space;" and "determining if the external location is within a second domain that is within a protection view of the first domain, *wherein the second domain owns the other one of the kernel space and a portion of the user space*."

The Examiner rejected claim 1 as being obvious in view of Kempf and in further view of Cannavino. (*See* 12/23/04 Office Action, p. 3). In support of this rejection, the Examiner points to Kempf's disclosure of an address space object which provides an interface for performing limited operations on the object's address space on behalf of client processes executing in non-supervisor mode. (*See* 12/23/04 Office Action, p. 3, ¶ 5) More generally, a program code manager provides an object oriented interface to a client process, thereby facilitating access to a program code segment object comprising executable binary code of a program. (*See* Kempf, Col. 6, ll. 42-46). This process may be categorized as one in which a user application interacts with another user application. However, it may not be included in the category in which a user application interacts with a kernel space, because such a process according to Kempf would require switching into supervisor mode. In contrast, the present invention, as recited in amended claim 1, would fall into such category where a user application interacts with a kernel space. The first domain, owning the memory space into which a code module is loaded, owns either a user space or a kernel space. The second domain, which may

own a location referenced by the code module loaded in the first domain, owns the other of the kernel space or user space that is not owned by the first domain. Accordingly, the present invention as claimed differs significantly from the teachings of Kempf, since the operations of each respective process occur within differing memory locations.

The Examiner also acknowledges that Kempf fails to disclose a method of handling potential irregularities "including exception handling for unauthorized accesses and attachment of separate address spaces." (*See* 12/23/04 Office Action, p. 4, ¶ 7). However, the Examiner contends that these deficiencies are cured by Cannavino. (*Id.*). Specifically, the Examiner points to Cannavino's disclosure of a method by which a problem program executing in a first address space obtains access to data in a second address space by executing a Set Second ASN ("SSAR") instruction. (*See* 12/23/04 Office Action, p. 3, ¶ 6). In this method, each address space has an associated set of address translation tables. (*See* Cannavino, Col. 15, ll. 66-68). These tables are used to locate address space control parameters, as well as a third authority table used to perform authorization tests. (*See* Cannavino, Col. 11, ll. 48-54). Primary and secondary table descriptors, which point to the first and second address spaces respectively, are stored in their respective control registers. (*See* Cannavino, Col. 15, l. 62 - Col. 16, l. 3). As the program executing in the first address space attempts to access data in the second address space, the descriptor pointing to the second address space is stored into the secondary control register. (*See* Cannavino, Col. 15, l. 68 - Col. 16, l. 3). The address in the secondary control register may then be compared to the address in the primary control register to determine if an address translation operation must be performed to obtain data in the other address space. (*See* Cannavino, Col. 16, ll. 3-10).

Amended claim 1 of the Applicants' invention recites "determining if the external location is within a second domain *that is within a protection view of the first domain*, wherein the second domain owns the other one of the kernel space and a portion of the user space." The Examiner acknowledges that neither Kempf nor Cannavino reference a protection view. (*See* 12/23/04 Office Action, p. 4, ¶ 7). However, the Examiner equates the protection view of the present invention with the address space tables described in Cannavino. (*Id*). Each protection domain of the present invention includes a mechanism allowing tasks executing in one protection domain to access resources and objects in a separate protection. (*See* Application, p.11, ll. 5-7).

One such mechanism is a protection view, included in each protection domain, which defines system resources and objects to which it has access. (*See* Application, p.11, ll. 7-9). A second domain may be found in the protection view of a first domain if the first domain has been attached to the second domain. (*See* Application, p.11, ll. 11-14). In contrast, the address space tables as described in Cannavino and mentioned above, is merely a reference which indicates the particular virtual address corresponding to an ASN. Each address translation table does not, however, specify the objects that are accessible by the domain by which the table is owned. Accordingly, the address translation table of Cannavino is not equivalent to the protection view claimed in the present invention.

Kempf does not teach or suggest all the limitations of amended claim 1, and Cannavino fails to cure these deficiencies. Accordingly, Applicants respectfully request that the rejection of claim 1 be withdrawn. Because claims 2-3 and 5 depend from and therefore include all the limitations of claim 1, it is respectfully submitted that these claims should also be allowed.

The Examiner rejected claim 8 as unpatentable over Kempf in view of Cannavino. (*See* 12/23/04 Office Action, pp. 4-5, ¶¶ 8-9). Claim 8, as amended, recites similar limitations to those of amended claim 1. Specifically, claim 8 recites a method comprising "creating a task in a first domain, *wherein the first domain owns one of a kernel space and a portion of a user space*, the task executing a number of instructions; and executing a first jump instruction in the number of instructions that refers to a link stub corresponding to an external location in a second domain, *wherein the second domain owns the other one of the kernel space and a portion of the user space.*" Accordingly, it is respectfully submitted that claim 8 should also be allowed for at least the reasons set forth above with respect to claim 1. Because claims 9-10 and 13-14 depend from and therefore include all the limitations of claim 8, it is respectfully requested that the rejection of these claims also be withdrawn.

The Examiner rejected claim 15 as unpatentable over Kempf in view of Cannavino. (*See* 12/23/04 Office Action, pp. 5-6, ¶¶ 10-11). As amended, claim 15 recites "a system space having a number of memory locations, *wherein the system space includes a kernel space and at least one user space*; and a number of protection domains, *at least one of the number of protection domains owning a portion of the system space.*" To support this rejection, the Examiner points to the same disclosure of Kempf, cited above, that was used as a ground for

rejection of claims 1 and 8. (*See* 12/23/04 Office Action, pp. 5, ¶¶ 10). As previously mentioned, the system described in Kempf is one in which a user application interacts with another user application. Accordingly, Kempf does not disclose a system space, including a kernel space, wherein a protection domain owns a portion of the system space. Cannavino fails to cure this deficiency. Claim 15 further recites *"wherein each of the number of protection domains includes a protection view* defining a set of the number of protection domains to which unprotected access may be made." As mentioned above, Cannavino fails to disclose a protection view, or an equivalent thereof. Therefore, it is respectfully submitted that claim 15, along with claims 16-23 and 25-27 which depend therefrom, should be allowed.

As discussed above, Kempf does not teach or suggest all the limitations of independent claims 1, 8, and 15. Cannavino fails to cure the deficiencies described for Kempf. Because claims 2-3, 5, 9-10, 13-14, 16-23, and 25-27 depend from and, therefore, include all of the limitations of corresponding claims 1, 8, and 15, it is respectfully submitted that these claims are also allowable over the cited references.

CONCLUSION

In light of the foregoing, the applicants respectfully submit that all of the now pending claims are in condition for allowance. All issues raised by the Examiner having been addressed, a notice of allowance should be issued.

Respectfully submitted,

Dated: April 25, 2005

By: _____
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel: (212) 619-6000
Fax:(212) 208-6819